Übungsblatt 6
„Mustererkennung“

J. Cavojska, N. Lehmann, R. Toudic

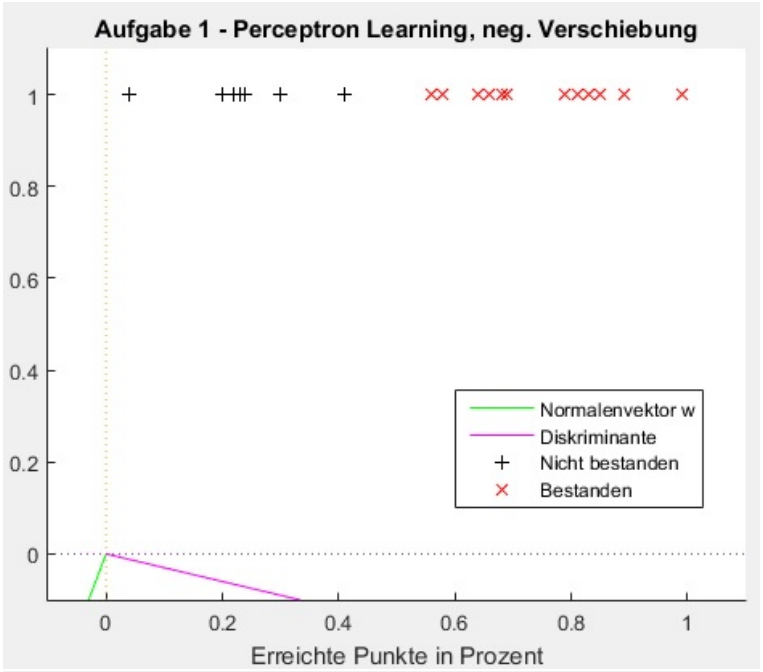
10.06.2015

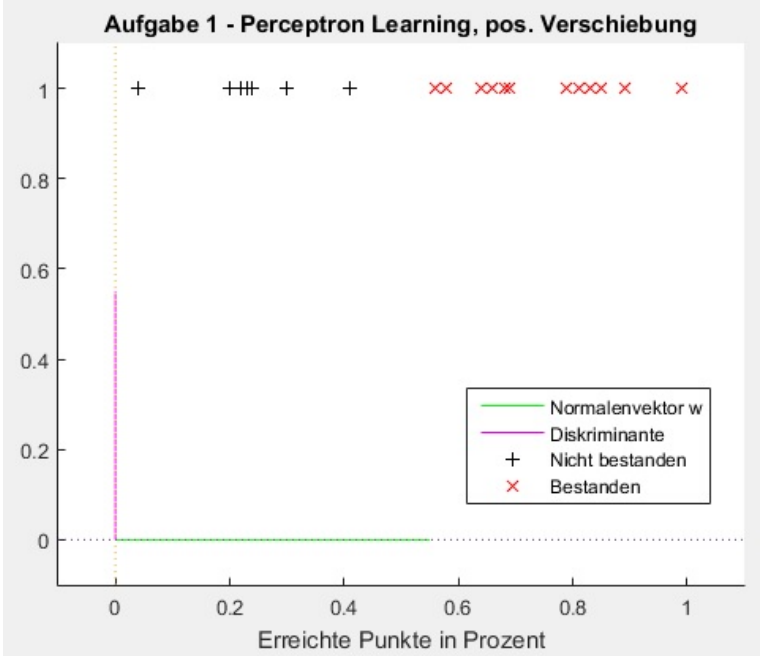
Inhaltsverzeichnis

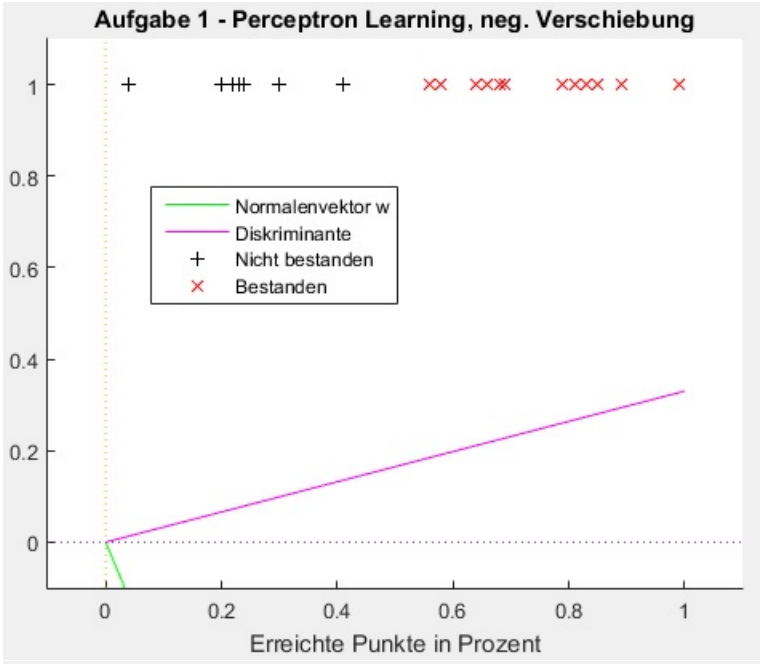
1 Perzeptron Lernalgorithmus

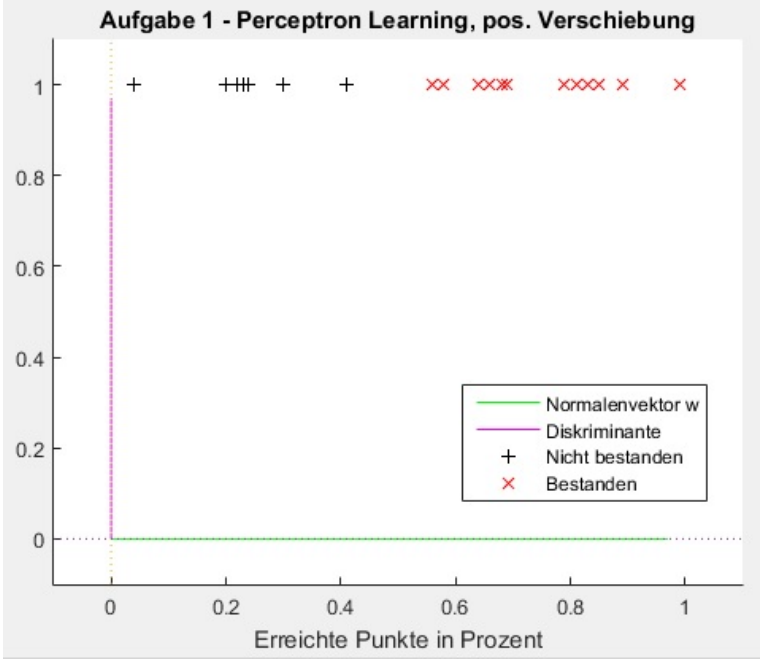
```
% Clean up
clear all
close all
clc

% Datenaufbereitung
Data     = load('klausur.txt');
Punkte   = Data(:,1);
Features = horzcat(Punkte, ones(size(Punkte,1), 1));
Noten    = Data(:,2);
Punkte0  = Data((Data(:,2)==0),:);
Punkte1  = Data((Data(:,2)==1),:);
x1       = linspace(0,1);
x2       = linspace(-5,5);


w = [0  0]; % random choosen vector w
limit = size(Data, 1);  % number of iterations | Abbruchkriterium
for i = 1:limit
    if w(1) == 0 && w(2) == 0
        w_norm = [0  0];
    else
        w_norm = w / norm(w);
    end
    lineNum = mod(i, size(Features,1))+1;
    proj = Features(lineNum, :) * w_norm'; % scalar projection

    if Noten(lineNum) == 1
        if proj < 0 % wrong classification

            Features(lineNum, :);
            w = w + Features(lineNum, :);
            w_norm = w / norm(w);
            w_x = w(1) * x1;

            if w(2) == 0
                w_y = w_x * 0;
            else
                coeff_w = w(2) / w(1)
                w_y = w_x * coeff_w;
            end

            diskriminante = [-w(2) w(1)];
            diskriminante_x = diskriminante(1) * x1;

            if diskriminante(1) == 0
                diskriminante_y = linspace(0, diskriminante(2));
            else
                coeff_d = diskriminante(2) / diskriminante(1);
                diskriminante_y = diskriminante_x * coeff_d;
            end

```

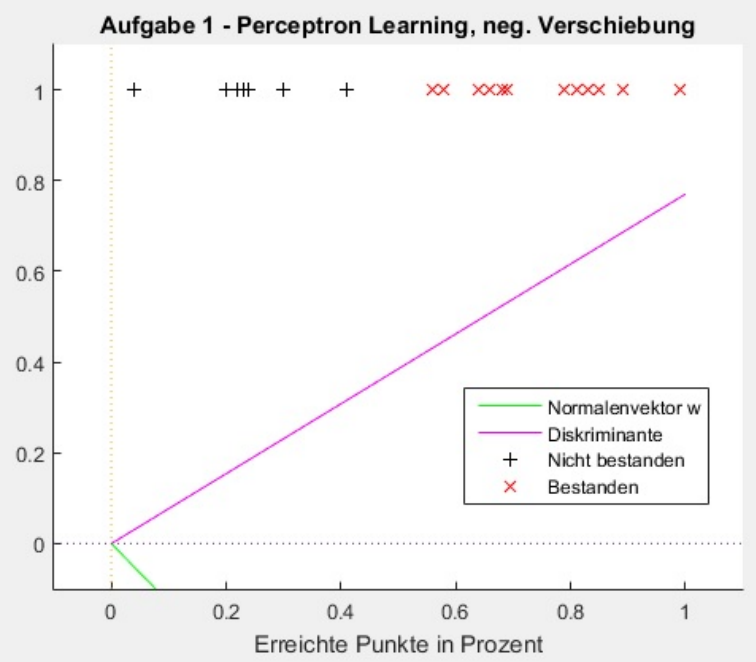
```
53              % plot
54              figure('NumberTitle','off','Name','Aufgabe 1 - Perceptron ↩
                    Learning');
55              hold on
56
57              plot(w_x, w_y, 'g');
58              plot(diskriminante_x, diskriminante_y, 'm');
59              gscatter(Punkte, ones(size(Punkte,1), 1), Noten,'krb','+x',[],'↩
                    off');
60
61              title('Aufgabe 1 - Perceptron Learning, pos. Verschiebung');
62              xlabel('Erreichte Punkte in Prozent');
63              axis([-0.1 1.1 -0.1 1.1]);
64              legend('Normalenvektor w', 'Diskriminante', 'Nicht bestanden','↩
                    Bestanden');
65              xL = xlim;
66              yL = ylim;
67              plot([0 0], yL, ':');
68              plot(xL, [0 0], ':');
69          end
70      end
71      if Noten(lineNum) == 0
72          if proj >= 0 % wrong classification
73
74              w = w - Features(lineNum,:);
75              w_norm = w / norm(w);
76              coeff_w = w(2) / w(1);
77              w_x = w(1) * x1;
78              w_y = w_x * coeff_w;
79              diskriminante = [-w(2) w(1)];
80              coeff_d = diskriminante(2) / diskriminante(1);
81              diskriminante_x = diskriminante(1) * x1;
82              diskriminante_y = diskriminante_x * coeff_d;
83
84              % plot
85              figure('NumberTitle','off','Name','Aufgabe 1 - Perceptron ↩
                    Learning');
86              hold on
87
88              plot(w_x, w_y, 'g');
89              plot(diskriminante_x, diskriminante_y, 'm');
90              gscatter(Punkte, ones(size(Punkte,1), 1), Noten,'krb','+x',[],'↩
                    off');
91
92              title('Aufgabe 1 - Perceptron Learning, neg. Verschiebung');
93              xlabel('Erreichte Punkte in Prozent');
94              axis([-0.1 1.1 -0.1 1.1]);
95              legend('Normalenvektor w', 'Diskriminante', 'Nicht bestanden', '↩
                    Bestanden');
96              xL = xlim;
97              yL = ylim;
98              plot([0 0], yL, ':');
99              plot(xL, [0 0], ':');
100         end
101     end
102 end
```

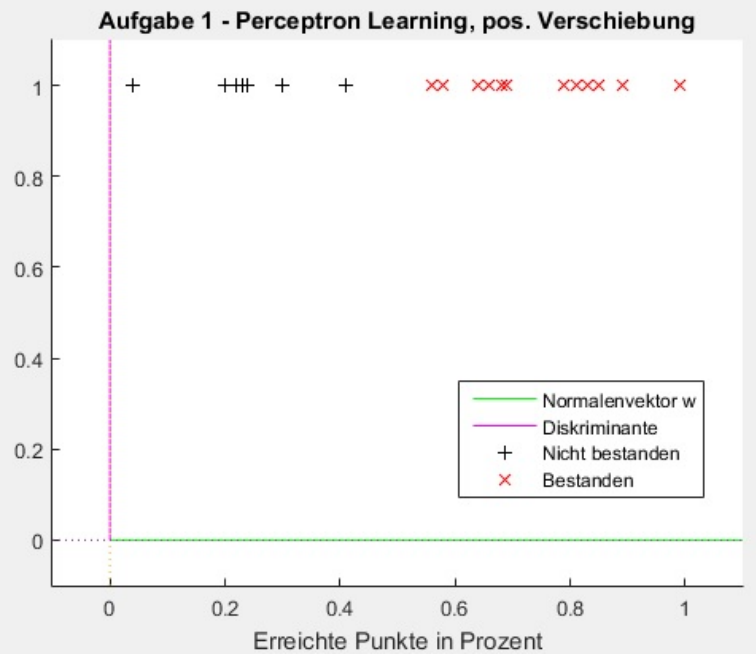

Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Erreichte Punkte in Prozent
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden

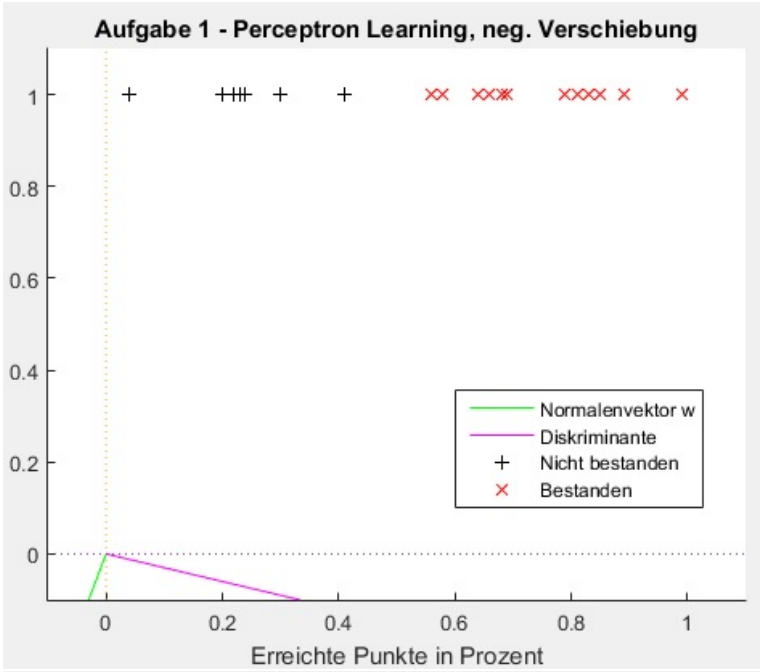

Aufgabe 1 - Perceptron Learning, pos. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Erreichte Punkte in Prozent
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden

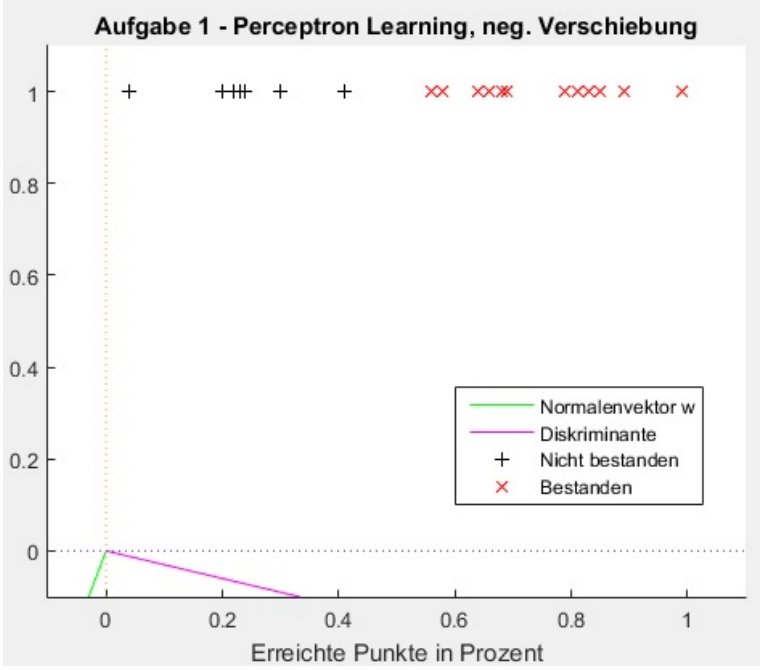

Aufgabe 1 - Perceptron Learning, neg. Verschiebung
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent

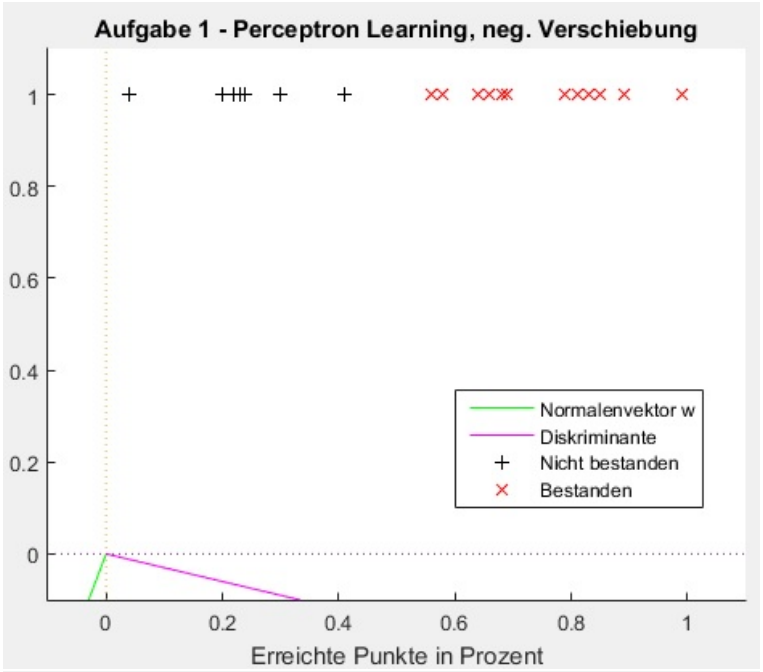

Aufgabe 1 - Perceptron Learning, pos. Verschiebung
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent

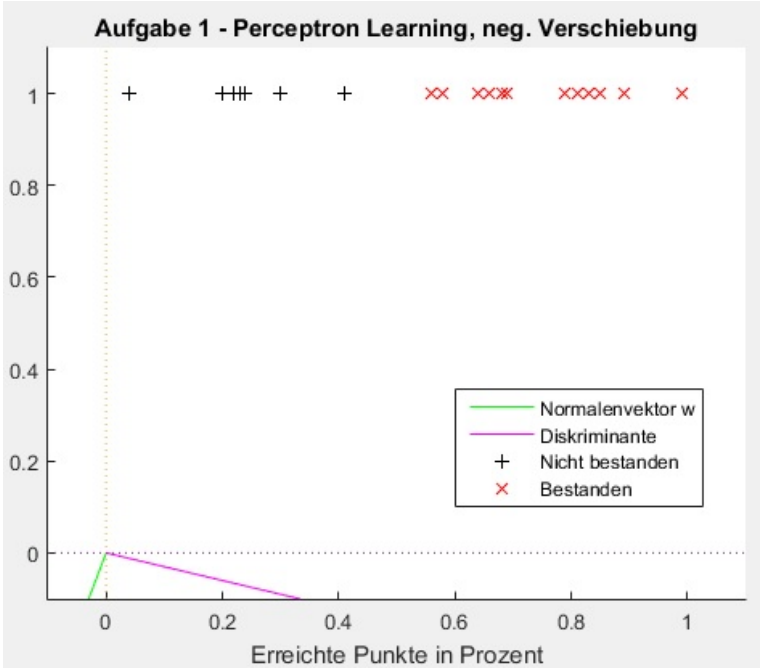

Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Erreichte Punkte in Prozent
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden


Aufgabe 1 - Perceptron Learning, pos. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Erreichte Punkte in Prozent
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden


Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent


Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent


Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent


Aufgabe 1 - Perceptron Learning, neg. Verschiebung
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Normalenvektor w
Diskriminante
Nicht bestanden
Bestanden
Erreichte Punkte in Prozent

2 Schwellwerte

2.1 Aufgabe 2A

```
schwellwerte = [];
for iter = 1:100
    randOrder = randperm(size(Features, 1));
    randFeatures = Features(randOrder');
    w = [max(Punkte) max(Noten)]; % random choosen vector w
    t = 0;
    limit = size(Data, 1); % number of iterations
    for i = 1:limit
        w_norm = w / norm(w);
        lineNum = mod(i, size(randFeatures,1))+1;
        proj = randFeatures(lineNum, :) * w_norm'; % scalar projection
        if Noten(lineNum) == 1
            if proj < 0
                t = t + 1;
                w = w + randFeatures(lineNum, :);
                diskriminante = [-w(2) w(1)];
            end
        end
        if Noten(lineNum) == 0
            if proj >= 0 % wrong classification
                t = t + 1;
                w = w - randFeatures(lineNum, :);
                diskriminante = [-w(2) w(1)];
            end
        end
    end
    schwellwerte = vertcat(schwellwerte, w);
end

% output
schwellwerte
mean_schwellwert = mean(schwellwerte)
```

2.2 Aufgabe 2B

```
figure('NumberTitle','off','Name','Aufgabe 2 - Lin. Regression');

% calculate
onesVector = ones(size(Data,1), 1);
X = horzcat(onesVector, Punkte);
beta = inv(X'*X) * X' * Noten;
fx = beta(1) + beta(2)*x2;
pkt = (0.5-beta(1))/beta(2);

% plot
hold on
scatter(Punkte, Noten, 'x', 'b')
plot (x2,fx, 'g')
scatter(pkt,0.5, 'o', 'r')

axis([-0.1 1.1 -0.1 1.1]);
legend('Noten', 'Diskriminante', 'Schwellenwert');
```